Assignment -2

1 Problem 1

In this problem we were given image of Wembley Stadium along with dimensions of Dee in the ground. We were asked to compute the length and width of the ground.

So to solve the the problem we compute homography matrix that would transform the given image into the image where the coordinates of the 4 corners of Dee are as $(0, 0)$ $(0, 18)$ $(44, 18)$ $(44, 0)$. After obtaining the homography matrix we can transform all points of the image in such a way such that the Dee have these coordinates. If the four vertices of the Dee indeed have these coordinates, then we can consider the image to be perpendicular to the principal axis (and such that the camera is far enough to view all the vertices without distortion). If the image is in this orientation, then the lengths of the field (in yards) is the same as the length (in pixels) that we get in the image plane.
Therefore, after we estimate the homography, we mark out three corners of the field (the fourth one is not visible in the image) and using the homography we estimated, we find its coordinates in the image plane of our choice. After this, calculating the width and length in yards is calculating the norm of the difference of the points, when represented in Euclidean coordinates.

How to Compute the Homography Matrix

In the Homography function, we take a bunch of points and the coordinates to where they are mapped and estimate a homography matrix that would implement such a transformation. The matrix is 3x3 and has 8 degrees of freedoms which means that we need at least 4 points. The actual estimation is done by stretching the matrix out into a vector of length 9 (Direct Linear Transform) and setting our estimation of this matrix to the last singular vector of the matrix A, where A is a 2*numberOfImagesx9 matrix containing values determined using the coordinates of the points in both planes.
In this problem, we use the fact that a homography can be used to model the transformation of a plane from one point of view to another.

2 Problem 2

In this we were required to find the width and height of the Wembley stadium without computing homographies. So we use cross ratios to find the actual length of the playing ground. We extend the Dee parallel to the width and mark the points as shown in the diagram below.

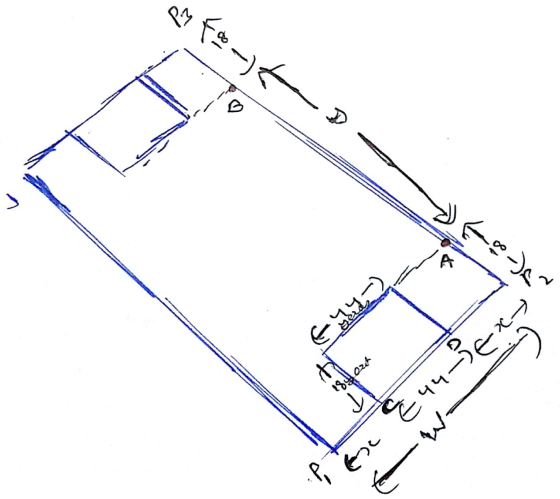


Figure 1: Image of the ground with construction

Note that image is not to scale

So to get length and width we get these equations :

$$crossratio1 = \frac{\frac{P1.D}{P1.P2}}{\frac{C.D}{C.P2}}$$

$$crossratio2 = \frac{\frac{P2.B}{P2.P3}}{\frac{A.B}{A.P3}}$$

As cross ratio remain invariant between homographies and perspective projections we get

$$crossratio1 = \frac{(x+44)^2}{44*(2*x+44)} = crossratio1-in-image-plane\ (usingpixels)$$

$$crossratio2 = \frac{(y+18)^2}{y*(2*y+18)} = crossratio2-in-image-plane\ (usingpixels)$$

And clearly $W = 2 * x + 44$ and $H = 2 * y + 18$.

On solving

$$\textbf{width} = \textbf{74.3241}$$

$$\textbf{length} = \textbf{115.5297}$$

Then we solve these equations and point of intersection of lines using the MATLAB symbolic mathematics toolbox. In the code for the file user can directly see the equations mentioned above to get the width and length.

On running myMainScript.m you get the value of length and width as output. Q2data.mat stores the pixel values for corners of ground, and corners of the 2 Dee's which we obtained by using the readPoints.m script that stores the location of points once clicked.

3 Problem 3

The Code

We have made functions to add noise to the image, calculate the joint entropies of two images as well as to rotate and translate the images. To rotate and translate the images, we use the imrotate and imtranslate methods, that use the nearest neighbour method by default. We had originally written a matrix to implement rotation and translation (using interp2), but using the functions was considerably faster. The jointEntropy functions computes the joint entropy as defined in the class notes.

The Result

In both cases the joint entropy was minimum for a rotation of 23 degrees, but the optimal translation was 0 and 1 pixels in the case of the barbara and flash images respectively. However, the gradient along the translation axis was very small and this could explain the results. The plots for both are as follow:

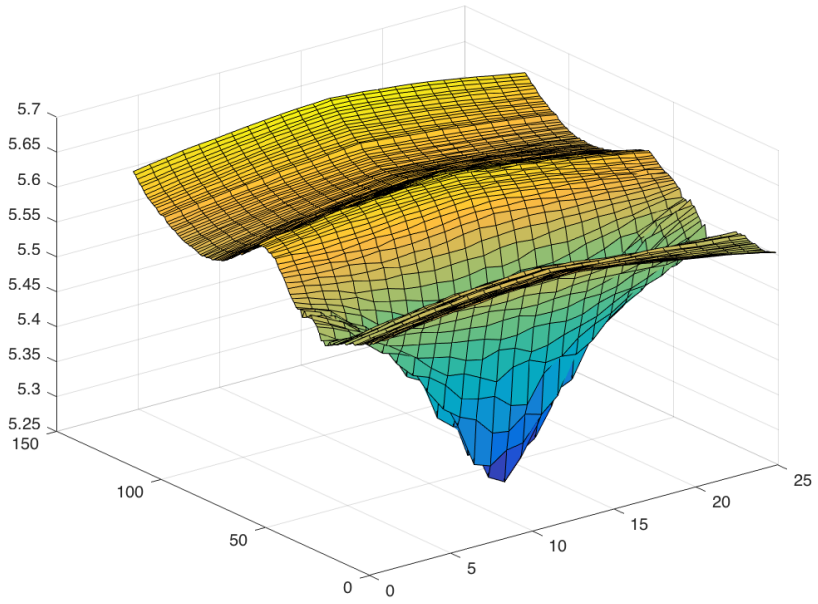


Figure 3: Plot for barbara image. Please subtract 61 and 13 from the labels of the axes to get the angle and the translation is actually corresponds to

The quality of alignment is better for the flash image, because the absolute

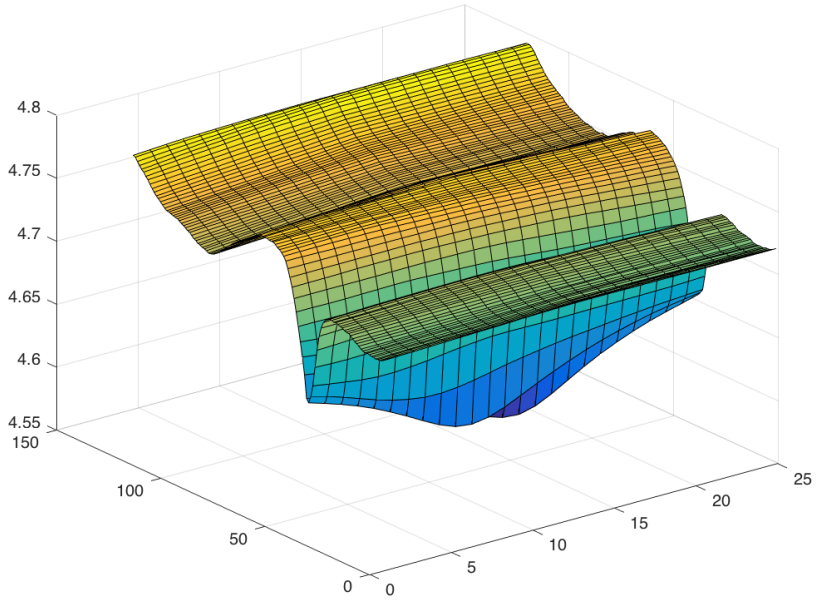


Figure 4: Plot for flash image. Please subtract 61 and 13 from the labels of the axes to get the angle and the translation is actually corresponds to

value for the joint entropy is lower in its case, despite the fact that the flash image is much bigger than the barbara one (around 9 times as many pixels).
On the other hand, we also observed that the gradient for the barbara image is more pronounced, that is, there is a clear and distinct minima, which is pretty localised (has small size along both axes) for which the value falls to around 5.2 from a maximum of 5.7 in the case of the barbara image. For the flash image, the minima has gentle gradient and is more spread out. Besides, the value falls to only to 4.55 from 4.8, which is a lesser percentage difference.

The 2-D surface plots for both the images are in the output folder, along with Q3Output.mat, which has the entropy matrices for both the images.
We have attached plots for all orientations (from -180 to 180 degrees and -256 to 256 pixels translation) and we found that the entropy was minimum when the image has been translated by close to 256 (or -256) pixels. In this case, the image has been completely translated out of the frame and the entire moving image is just black. This is what reduces the joint entropy greatly as the probability distribution for the moving image is a delta function. This means that in the joint histogram, only the row corresponding to the bin 0-9 for the moving image will have non-zero entries and the more concentrated the joint

distribution, the lesser the joint entropy.

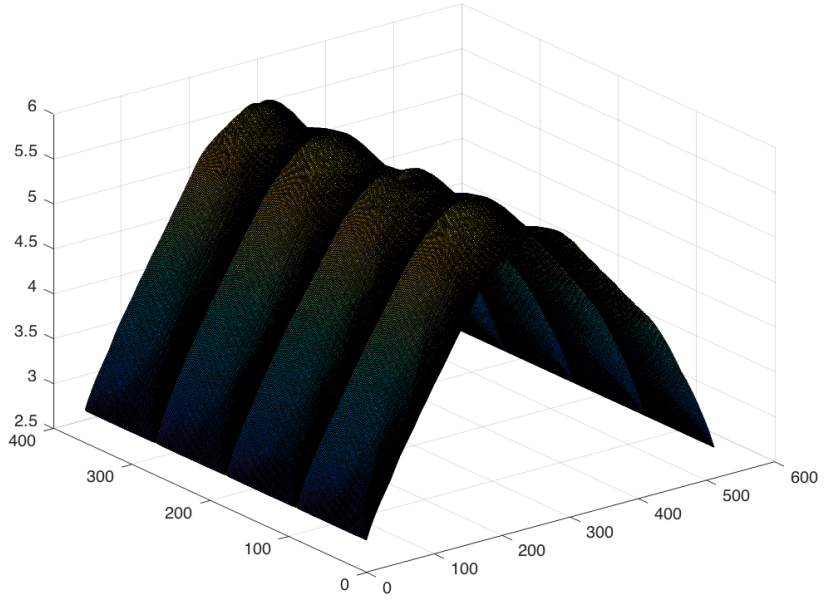


Figure 5: Plot for all orientation of the barbara image. The joint entropy tapers an falls close to either of the ends for the translation axis. Please subtract 181 and 257 from the labels of the axes to get the angle and the translation is actually corresponds to

4 Problem-4

We built the function ransacHomography to compute homography matrix from a list of points (selecting points at random, conputing the homography and then taking the one that maximises the consensus set) and then warped the images using the homographies to warp and thus stitch the images.
Note that our code is compatible with MATLAB 2016 and later versions.

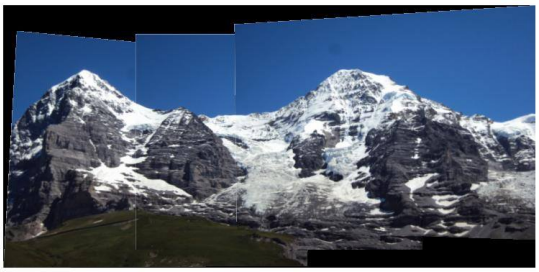


Figure 6: Image obtained on stitching images of hills

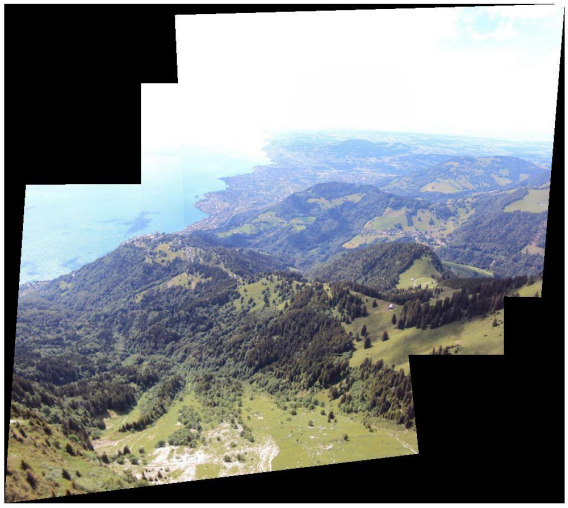


Figure 7: Image obtained on stitching images of ledge

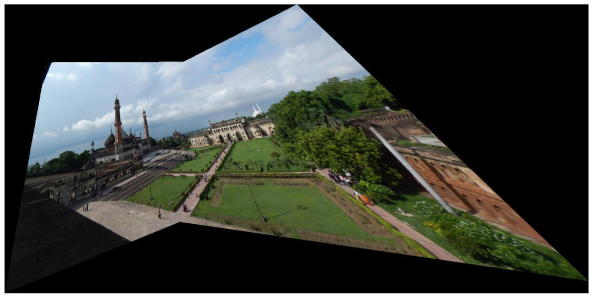
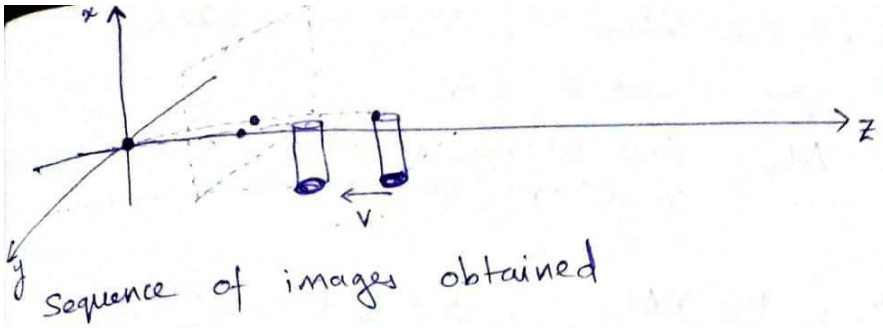


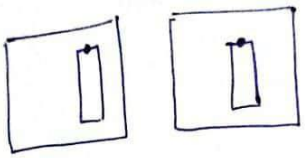
Figure 8: Image obtained on stitching images of monument

Figure 9: Image obtained on stitching images of pier

5 Problem-5



Sequence of images obtained



Let the camera capture images at an interval Δt_0

Consider the center-point of the top horizontal line in every image

Let the initial co-ordinate (world-co-oridinate) of the corresponding point be (x_0, y_0, z_0) and at other captured time instances be (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), ...

Let the corresponding image co-ordinates in different images of the sequence be (u_0, v_0), (u_1, v_1), (u_2, v_2) ....

then we know that

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \frac{f}{z_i}$$

$$u_0 = \frac{x_0 f}{z_0}, \quad u_1 = \frac{x_1 f}{z_1}, \quad u_2 = \frac{x_2 f}{z_2}$$

$$\therefore \quad \frac{u_0}{u_1} = \frac{x_0}{x_1}\frac{z_1}{z_0}, \quad \frac{u_0}{u_2} = \frac{x_0}{x_2}\frac{z_2}{z_0}, \quad \ldots$$

Since bar is moving along z-axis $x_0 = x_1 = x_2 \ldots$

$$\therefore \quad \frac{u_0}{u_1} = \frac{z_1}{z_0}, \quad \frac{u_0}{u_2} = \frac{z_2}{z_0}, \quad \ldots$$

now, $Z_1 = Z_0 - V\Delta t_0$, $Z_2 = Z_0 - 2V\Delta t_0$

where V = speed of bar

Δt_0 = time interval at which camera registers images.

$$\therefore \frac{U_0}{U_1} = 1 - \frac{V\Delta t_0}{Z_0} \quad , \quad \frac{U_0}{U_2} = 1 - \frac{2V\Delta t_0}{Z_0}$$

$$\frac{U_0}{U_3} = 1 - \frac{3V\Delta t_0}{Z_0} \quad \cdots$$

$$\therefore \frac{V\Delta t_0}{Z_0} = \frac{U_0}{U_1} - 1 \Rightarrow \frac{\Delta t_0}{\left(\frac{Z_0}{V}\right)} = \frac{U_0}{U_1} - 1$$

Now, since V is const. and Z_0 is the original distance of bar from the plane $\perp$ to optical axis and passing through camera origin point.

the time required is $\frac{Z_0}{V}$

we can get U_0, U_1 from the first and second frame respectively

then $$\boxed{\frac{Z_0}{V} = \frac{\Delta t_0}{\left(\frac{U_0}{U_1}\right) - 1}}$$

(Since we know the time gap, U_0, U_1)